PATENT ABSTRACTS OF JAPAN

(11)Publication number : 2001-022808
(43)Date of publication of application : 26.01.2001

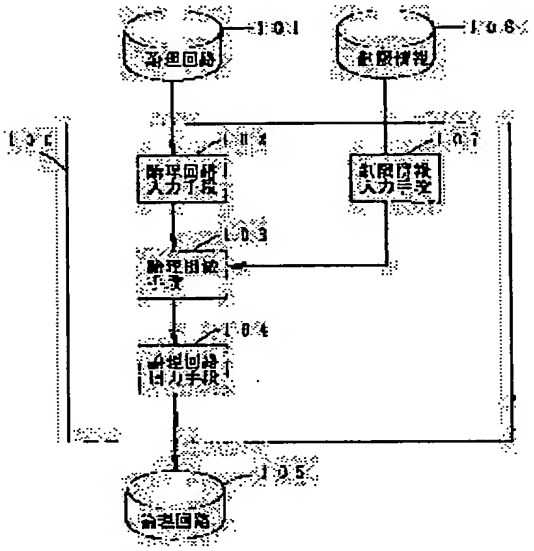
(51)Int.Cl. G06F 17/50
G01R 31/28

(21)Application number : 11-193869
(22)Dat of filing : 08.07.1999

(71)Applicant : MATSUSHITA ELECTRIC IND CO LTD
(72)Inventor : MAE YOICHIRO
KAWAMOTO ISAO

(54) LOGIC CIRCUIT REDUCING DEVICE, METHOD AND DEVICE FOR LOGIC SIMULATION

(57)Abstract:
PROBLEM TO BE SOLVED: To reduce logic corresponding to the purpose of verification at a high speed in the logic verification of a large scale circuit.
SOLUTION: Concerning this logic circuit reducing device, a device 100 for converting logic circuits is provided with a means 102 for inputting an object circuit to be verified composed of a logic circuit 101, a limit information input means 107 for supplying limit information 106 corresponding to the verification purpose and a logic reducing means 103 for reducing the inputting object circuit to be verified on the basis of the limit information. The logic simulation method simulates the operation of the logic circuit 101 in which the logic is reduced using this logic circuit reducing device.



LEGAL STATUS

[Dat of request for examination]

[Date of sending the examiner's decision of rejection]

[Kind of final disposal of application other than the examiner's decision of rejection or application converted registration]

[Date of final disposal for application]

[Pat nt number]

[Dat of registration]

[Number of appeal against examiner's decision of rejection]

[Date of r questing appeal against examiner's decision of rejection]

[Dat of xtinction of right]

(19)日本国特許庁（ＪＰ）　(12)公開特許公報（Ａ）　(11)特許出願公開番号
特開2001－22808
（Ｐ2001－22808Ａ）
(43)公開日　平成13年1月26日(2001.1.26)

(51)Int.Cl.⁷	識別記号	ＦＩ		ﾃｰﾏｺｰﾄﾞ(参考)
Ｇ06Ｆ 17/50		Ｇ06Ｆ 15/60	664Ｋ	2Ｇ032
Ｇ01Ｒ 31/28		Ｇ01Ｒ 31/28	Ｆ	5Ｂ046

審査請求　未請求　請求項の数5　ＯＬ　（全25頁）

(21)出願番号　特願平11－193889

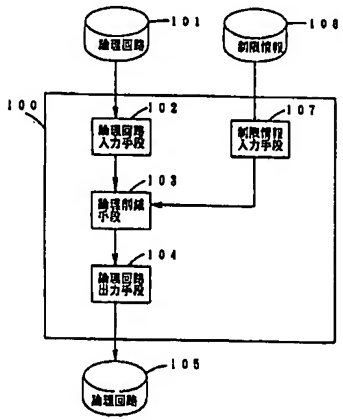
(22)出願日　平成11年7月8日(1999.7.8)

(71)出願人　000005821
松下電器産業株式会社
大阪府門真市大字門真1006番地
(72)発明者　前　祥一郎
大阪府門真市大字門真1006番地　松下電器産業株式会社内
(72)発明者　河本　功
大阪府門真市大字門真1006番地　松下電器産業株式会社内
(74)代理人　100076174
弁理士　宮井　暎夫

Ｆターム(参考)　2Ｇ032 ＡＡ01 ＡＣ03 ＡＨ07
5Ｂ046 ＡＡ08 ＢＡ03 ＪＡ05

(54)【発明の名称】　論理回路削減装置ならびに論理シミュレーション方法および装置

(57)【要約】
【課題】大規模回路の論理検証において、検証目的に応じた論理の削減を行い、高速な論理回路削減装置ならびに論理シミュレーション方法および装置を提供する。
【解決手段】論理回路１０１から構成される被検証対象回路を入力する手段と、検証目的に応じた制限情報１０８を与える制限情報入力手段１０７と、制限情報に基づき、入力された被検証対象回路を削減する論理削減手段１０３を備える。論理シミュレート方法は、この論理回路削減装置を用いて論理を削減された論理回路の動作をシミュレートする。



1

【特許請求の範囲】
【請求項１】　論理回路から構成される被検証対象回路を入力する手段と、検証目的に応じた制限情報を与える制限情報入力手段と、前記制限情報に基づき、入力された前記被検証対象回路を削減する論理削減手段を備えた論理回路削減装置。
【請求項２】　論理回路から構成される被検証対象回路の動作のシミュレーション装置であって、請求項１記載の論理回路削減装置を用いて論理を削減された論理回路の動作をシミュレートすることを特徴とする論理シミュレーション方法。
【請求項３】　論理回路から構成される被検証対象回路の動作のシミュレーション装置であって、検証目的に応じかつシミュレーション時刻に依存した制限情報を与える制限情報入力手段と、前記シミュレーション時刻に依存した前記制限情報に基づき前記シミュレーション時刻を分割するシミュレーション時刻分割手段と、分割されたシミュレーション時刻毎に入力された前記被検証対象回路を削減する論理回路削減手段とを備え、削減された論理回路の動作をシミュレートすることを特徴とする論理シミュレーション装置。
【請求項４】　プロセッサ部と、論理回路部とから成る被検証対象回路の動作の論理シミュレーション装置であって、前記論理回路部の制限可能情報を与える可制限情報入力手段と、前記プロセッサ部を制御するソフトウェアの命令動作を解析する手段と、前記ソフトウェアの命令動作の解析により制限情報を抽出する制限情報抽出手段と、抽出された前記制限情報に基づき前記被検証対象回路の前記論理回路部を削減する論理削減手段とを備え、前記プロセッサ部の命令動作まで含めて前記論理回路部の動作をシミュレートすることを特徴とする論理シミュレーション装置。
【請求項５】　プロセッサ部と、論理回路部とから成る被検証対象回路の動作の論理シミュレーション装置であって、前記プロセッサ部を制御するソフトウェアの検証範囲を指定するソフトウェア指定情報入力手段と、前記論理回路部の制限可能情報を与える可制限情報入力手段と、前記ソフトウェア指定情報入力手段のソフトウェア指定情報と前記論理回路部の可制限情報より制限情報を生成し且つ前記プロセッサ部のプログラム命令系列を生成するコンパイラと、抽出された前記制限情報に基づき前記論理回路部を削減する論理削減手段とを備え、前記プロセッサ部の命令動作まで含めて前記論理回路部の動作をシミュレートすることを特徴とする論理シミュレーション装置。
【発明の詳細な説明】
【０００１】
【発明の属する技術分野】本発明は、論理を削減する論理回路削減装置ならびに論理の検証のための論理シミュレーション方法および装置に関するものであり、論理回路の動作のシミュレーションを高速にシミュレートする装置を提供するものである。

2

【０００２】
【従来の技術】従来の論理検証は、被検証対象回路である論理回路に対する論理シミュレータによるシミュレーションによって行なわれている。多くの論理シミュレーション処理は主にイベントドリブン処理で行なわれている。また、同期回路のクロックサイクルにイベントを限定した、高速なサイクルベースに基づく論理シミュレーション手法も開発され、実用化されている。
【０００３】また、被検証対象回路を、入力テストパターンの解析により信号値が変化しない部分の論理を削減することにより、被検証対象回路のシミュレーションを高速化する方法が開示されている（特開平６－２４３１９０号）。
【０００４】また、プロセッサ部とプロセッサにより制御されるハードウェアを持ち、ソフトウェアプログラムコードとハードウェアが相互に作用を及ぼす動作の検証方法としては、プロセッサ部を命令レベルシミュレータでシミュレートし、論理回路部をクロックサイクルにおいて実際のチップと同一動作をする（以後、サイクルアキュレート）シミュレーションモデルから、プロセッサからの入出力アクセス毎の動作レベルに変換し、プロセッサ部と変換した論理回路部によりシミュレーションを行なう方法が開示されている（特開平１０－１８７７８９号）。
【０００５】
【発明が解決しようとする課題】通常の論理シミュレーションでは、非常に大規模な論理回路の場合、論理検証に膨大な時間がかかる。半導体技術の進歩により１つの半導体集積回路の中にプロセッサ部、論理回路部、メモリがまとめられる現状においては、サイクルベースシミュレータを用いたとしても、膨大な検証が必要であり、現実的な時間で論理検証を行なうことは難しい。また、このような大規模回路の検証は、複数の検証パターンに分割されることが多く、各検証パターンでは論理回路の一部しか動作しないことが多く、余分なシミュレーションを行なう結果となる。
【０００６】また、被検証対象回路に与える入力テストパターンの解析により、信号値が変化しない部分の論理を削減することによりシミュレーション対象となる論理回路の削減を行い、シミュレーションの高速化を図る方法では、余分なシミュレーションを行なうことはないが、論理の削減が入力テストパターンに依存しているため、そのテストパタン作成は容易ではない。さらに、被検証対象回路内にメモリやレジスタが存在する場合、シミュレーションの初期において値を設定する。入力テストパターン上では、初期設定後にメモリやレジスタの値を変更しない場合、入力テストパターンに依存した情報からではメモリやレジスタの削除を行なうことはできな

い。さらに、入力テストパターン全体に依存して論理を削減するために、入力ベクタの一部で変化して、大部分で変化しないことが分かっている場合でも論理を削除することができない。

【0007】プロセッサ部を命令レベルシミュレータでシミュレートし、論理回路部をプロセッサからの入出力アクセス毎の動作レベルに変換する方法では、論理回路部の抽象度が高いために高速シミュレーションが可能であるが、クロックサイクルでの検証が行なわれておらず、検証精度に問題がある。

【0008】本発明は、かかる点に鑑みてなされたものであり、その目的は、被検証対象論理回路の論理検証の論理シミュレーションにおいて、検証項目を限定し、その限定情報に基づいて論理を削減し、高速に論理シミュレーションを行なうことができる論理回路削減装置および論理シミュレーション装置を提供するものである。

【0009】また、プロセッサ部と論理回路部を含み、ソフトウェアプログラムコードとハードウェアが相互に作用するシステム全体のシミュレーションにおいて、論理回路部の情報と、ソフトウェアの解析により、検証を行なうソフトウェアから論理回路部の削減情報を作成し、論理を削減することにより高速な論理シミュレーション装置を提供するものである。

【0010】

【課題を解決するための手段】上記目的を達成するために、大規模論理回路の論理シミュレーション装置において、本発明は、被検証対象回路に対して検証目的に応じた制限情報を与え、与えた制限情報に基づき、被検証対象回路の論理を削減する論理回路削減装置であり、この論理削減装置を用いてシミュレートすることにより、論理シミュレーションの速度を高速化し、検証時間を削減する論理シミュレーション方法および装置である。

【0011】請求項1記載の論理回路削減装置は、論理回路から構成される被検証対象回路を入力する手段と、検証目的に応じた制限情報を与える制限情報入力手段と、制限情報に基づき、入力された被検証対象回路を削減する論理削減手段を備えたものである。

【0012】請求項1記載の論理回路削減装置によれば、検証目的に応じた制限情報を与え、その制限情報に基づき論理回路から構成される被検証対象回路を削減する方法により、被検証対象回路を入力し、検証目的に応じた制限情報を入力し、入力した制限情報に基づき、入力された被検証対象回路を削減することができる。このため、高速に論理シミュレーションを行なうことができる。

【0013】請求項2記載の論理シミュレーション方法は、論理回路から構成される被検証対象回路の動作のシミュレーション装置であって、請求項1記載の論理回路削減装置を用いて論理を削減された論理回路の動作をシミュレートすることを特徴とするものである。

【0014】請求項2記載の論理シミュレーション方法によれば、削減された論理回路の動作をシミュレートすることにより、論理シミュレーションを高速化することができる。

【0015】請求項3記載の論理シミュレーション装置は、論理回路から構成される被検証対象回路の動作のシミュレーション装置であって、検証目的に応じかつシミュレーション時刻に依存した制限情報を与える制限情報入力手段と、シミュレーション時刻に依存した制限情報に基づきシミュレーション時刻を分割するシミュレーション時刻分割手段と、分割されたシミュレーション時刻毎に入力された被検証対象回路を削減する論理回路削減手段とを備え、削減された論理回路の動作をシミュレートすることを特徴とするものである。

【0016】請求項3記載の論理シミュレーション装置によれば、検証目的に応じた制限情報の各々にシミュレーション時刻に依存した制限情報を与え、シミュレーション時刻に依存した制限情報に基づき、シミュレーション時刻を分割し、分割されたシミュレーション時刻毎に被検証対象回路の論理を削減し、論理シミュレートする方法により、全体の論理シミュレーションを高速化することができる。

【0017】請求項4記載の論理シミュレーション装置は、プロセッサ部と、論理回路部とから成る被検証対象回路の動作の論理シミュレーション装置であって、論理回路部の制限可能情報を与える可制限情報入力手段と、プロセッサ部を制御するソフトウェアの命令動作を解析する手段と、ソフトウェアの命令動作の解析により制限情報を抽出する制限情報抽出手段と、抽出された制限情報に基づき被検証対象回路の論理回路部を削減する論理削減手段とを備え、プロセッサ部の命令動作まで含めて論理回路部の動作をシミュレートすることを特徴とするものである。

【0018】請求項4記載の論理シミュレーション装置によれば、プロセッサ部を制御するソフトウェアの命令動作を解析し、論理回路部に与えることができる制限情報を入力し、この制限情報とソフトウェアの命令動作の解析結果から、論理を削減する制限情報を抽出し、抽出された制限情報に基づき論理を削減する方法により、論理シミュレーションを高速化することができる。

【0019】請求項5記載の論理シミュレーション装置は、プロセッサ部と、論理回路部とから成る被検証対象回路の動作の論理シミュレーション装置であって、プロセッサ部を制御するソフトウェアの検証範囲を指定するソフトウェア指定情報入力手段と、論理回路部の制限可能情報を与える可制限情報入力手段と、ソフトウェア指定情報入力手段のソフトウェア指定情報と論理回路部の可制限情報より制限情報を生成し且つプロセッサ部のプログラム命令系列を生成するコンパイラと、抽出された制限情報に基づき論理回路部を削減する論理削減手段とを備え、プロセッサ部の命令動作まで含めて論理回路部の動作をシミュレートすることを特徴とするものである。

【0020】請求項5記載の論理シミュレーション装置によれば、検証目的に応じて、プロセッサ部を制御するソフトウェアを指定し、論理回路部に与えることができる制限情報から、プロセッサ部を制御するプログラム命令系列を生成するとともに論理回路部の制限情報を生成し、生成された制限情報に基づき論理を削減する方法により、シミュレーションを高速化することができる。

【0021】

【発明の実施の形態】以下、本発明の実施の形態について、図を用いて説明する。

【0022】（第1の実施の形態）第1の実施の形態では、請求項1に係る発明に関して図を用いて説明する。

【0023】図1は請求項1に係る論理回路削減装置の構成を示す図である。

【0024】図1において論理回路の変換装置100は、変換する対象の回路101を入力する論理回路入力手段102と、論理回路101のシミュレーション実行時の制限情報106を入力する制限情報入力手段107と、制限情報106に基づいて対象の回路101から所望の検証に不必要な回路を削除する論理削減手段103と、不必要な回路を削除して得られた変換後の論理回路105を出力する論理回路出力手段104を備えている。

【0025】次に、例を用いて第1の実施の形態に係る論理回路削減の方法を具体的に説明する。図2は論理回路の削減の対象となるハードウェア記述言語を用いて記述された論理回路の記述の例である。

【0026】図3は図2の論理回路に対してシミュレーション時の使用方法について与える制限情報の一例である。

【0027】図3において300は制限を加える信号名、301は加える制限の種類、302は信号を固定して用いる場合の信号値である。

【0028】論理回路に加える制限の種類301において、"fix"は信号値を固定して用いることを示している。

【0029】この制限は、必要とする検証を行なう期間内には値を固定して用いる場合等に使用することができる。

【0030】また、"ignore"はシミュレーション時に値を参照する必要のない信号であることを示しており、必要とする検証において、値を観測する必要がない信号がある場合などに用いることができる。

【0031】図4は図2の論理回路を、図3の制限情報に基づき第1の実施の形態での論理回路削減装置を使用した結果作成される論理回路の記述の例である。

【0032】図23は第1の実施の形態に係るハードウェアの構成図の一例を示しており、図23において、2301は処理されたあらゆる情報をみるためのディスプレイ装置、2302は設計者があらゆる情報や処理命令を入力するためのキーボード、2303はあらゆる処理を行う中央演算処理装置、2304は各情報を格納する記憶装置である。

【0033】図24は論理削減手段103で実行する論理の削減方法を示す図である。図24において、2401は制限情報106に処理していない項目が存在するかどうかを判定する処理、2402は制限情報106から未処理の制限情報を選択する処理、2403は選択した制限情報により指定される信号を含む被代入項または式の中に未処理のものが存在するかどうかを判定する処理、2404は選択した制限情報により指定される信号を含む被代入項または式を選択する処理、2405は2404で選択した式を2402で選択した制限情報を用いて論理の圧縮を行なう処理、2406は2405で論理圧縮された式または2404で選択した被代入項を含む文を圧縮または削除する処理である。

【0034】2405の式の圧縮処理では、制限情報が信号の固定である場合に、選択した式中の指定信号を定数に置き換え、演算規則に従い論理を圧縮する。

【0035】2406の文の圧縮または削除処理では、条件式が固定値となった条件文の圧縮、被代入項の信号が固定する信号または観測する必要がない信号に指定されている場合の代入文の削除、文を削除することより内容のなくなった複文の削除を行なう。

【0036】次に、実際に図24の方法に従い、図2の論理回路の記述を図3の制限情報を用いて圧縮する手順について説明する。

【0037】図2の論理回路に対して図3の制限情報が与えられた場合、まず最初の制限情報303に注目する。

【0038】制限情報303はレジスタ変数"mode"を定数に固定する指定であるため、図2からレジスタ変数"mode"を含む式を検索し、選択した式に含まれる変数"mode"を定数に置き換え、論理を圧縮する。

【0039】また、信号を固定する変数が代入文の左辺に存在する場合、その文を削除する。

【0040】具体的には図2での30行から37行に対しては、各代入文の右辺の式に変数"mode"が含まれるため、これを固定値（16進数の"fa"）に置き換え、式を論理圧縮することにより、図4の21行から24行のように圧縮され、図2での20行と23行に対しては各代入文の被代入項が値を固定する信号であるため、その代入文を削除し、これにより1行とalways文の内容がなくなるためこれらも削除し結局18行から25行の文が削除される。

【0041】次に、第2の制限情報304に注目し、変数"re"が含まれる式を圧縮する。

【００４２】さらに、第３の制限情報３０５に注目する。

【００４３】制限情報３０５は変数"sigovf"を観測しないという指定であるため、"sigovf"の値をシミュレーション中に求める必要はない。

【００４４】このため、図２の４１行の"sigovf"を被代入項に持つ代入文を削除する。

【００４５】以上の処理により、論理回路図２の記述は図４に変換される。

【００４６】なお、ここでは入力とする論理回路として、Verilog-HDLで　かれたRTレベルの記述を用いた例を用いたが、VHDLやその他の記述言語、また動作レベルやゲートレベルの記述に対しても同様に不要部分の削除を行なえることは言うまでもない。

【００４７】以上のように、本発明の第１の実施の形態によれば、請求項１に係る発明である論理回路削減装置によって、シミュレーションの対象となる論理回路から、目的とする検証に不必要な部分を削除することにより、高速なシミュレーションが可能で、回路規模の小さいシミュレーションモデルの記述を得ることができる。

【００４８】（第２の実施の形態）第２の実施の形態では、請求項２、３に係る発明に関して図を用いて説明する。図５は請求項２、３に係るシミュレータの構成を示す図である。

【００４９】図５において論理回路のシミュレータ５００は、被検証対象回路５０１を入力する論理回路入力手段５０２と、論理回路に対するミュレーション実行時の制限情報５０７を入力する制限情報入力手段５０８と、静的な制限情報５０７に基づいて対象回路から所望の検証に不必要な回路をシミュレーション開始前に削除する第１の論理削減手段５０３と、動的な制限情報に基づいてシミュレーション時刻を分割するシミュレーション時刻分割手段５０９と、分割されたシミュレーション時刻毎に被検証対象回路（論理削減後の論理回路５０４）から不要部分を削除してシミュレーションモデルを作成する第２の論理回路削減手段５０５と、作成されたシミュレーションモデルを用いてシミュレーションを行なうシミュレーション手段５０６を備えている。

【００５０】次に、例を用いて第２の実施の形態に係るシミュレータを具体的に説明する。

【００５１】図８はシミュレーション対象となる論理回路により構成されるシステムモデルの最上位階層を表す記述の一部である。

【００５２】ただし、図８において、モジュール"filter"は、図２の記述で表された論理回路である。

【００５３】図７は図８のシステムモデルに対してシミュレーション時の使用方法について与える制限情報の一例である。

【００５４】図７において７００は制限を加える信号またはモジュール名である。

【００５５】７０１は加える制限の種類であり、"fix"は信号値を固定して用いることを示し、"ignore"はシミュレーション時に値を参照する必要のない信号であることを示し、"remove"はモジュールインスタンスを取り除くことを示している。

【００５６】７０２は信号を固定して用いる場合の信号値であり、"hold"は信号値を固定する時刻に設定されている値に固定することを示している。

【００５７】７０３は制限を加えるシミュレーション時刻を設定する条件、７０４は制限を解除するシミュレーション時刻を設定する条件であり、"initial"はシミュレーション開始前に制限を加えることを示し、"#1000"はシミュレーション時刻"1000"に制限を加えることを示し、"posedge system.mode"、"negedge system.mode"はそれぞれ信号"mode"の値が１または０に変化した時点で制限の設定、解除を行なうことを示している。

【００５８】図２５は、図５の５０５で実行する論理回路削減方法を示す図である。

【００５９】図２５において、２５０１は時刻分割手段５０９により分割された時刻に加える制限の中に未処理のものが存在するかを判定する処理、２５０２は制限情報５０７から処理を行なう制限を選択する処理、２５０３は選択した制限を用いてインスタンス内の論理回路を圧縮する処理、２５０４は選択した制限に従いインスタンスを削除する処理、２５０５は削除したインスタンスの出力ポートに接続していた信号線をインスタンスを削除した時点での値に設定する処理、２５０６は時刻分割手段５０９により分割された時刻に解除する制限の中に未処理のものが存在するかを判定する処理、２５０７は解除する制限を選択する処理、２５０８は選択した制限を用いて加えていた変更を元の状態に戻す処理である。

【００６０】ただし、２５０３の論理回路を圧縮する処理では第１の実施の形態で示した図２４に従い、論理の圧縮を行なう。

【００６１】図２３は第２の実施の形態に係るハードウェアの構成図の一例を示しており、図２３において、２３０１は処理されたあらゆる情報をみるためのディスプレイ装置、２３０２は設計者があらゆる情報や処理命令を入力するためのキーボード、２３０３はあらゆる処理を行う中央演算処理装置、２３０４は各情報を格納する記憶装置である。

【００６２】第２の実施の形態に係るシミュレータでは、被検証対象回路（図６）と制限情報（図７）が与えられると、制限情報（図７）の中から、シミュレーション開始前に処理可能な静的な制限情報を選び出し、第１の論理削減手段５０３により、第１の実施の形態と同様に回路を削減する。

【００６３】図７の例では、７０３に"initial"の設定がされ、７０４には何も設定されていない項目が選び出され、インスタンス"system.filter1"に対応するモジュールは図４のように変換され、インスタンス"system.filter2"に対応するモジュールは図８のように変換される。

【００６４】次に、第１の論理削減手段５０３により作成されたシミュレーションモデルを用いて、シミュレーション手段５０６によりシミュレーションを行なう。

【００６５】この時、第２の実施の形態に係るシミュレータでは、シミュレーション時刻分割手段５０９により、シミュレーションの実行時に制限情報に指定された条件７０３または７０４が成立していないかを観測し、成立している場合、第２の論理回路削減手段５０５により、被シミュレーション対象のモデルを動的な制限情報に基づいて図２５のように変換する。

【００６６】図７の例では、４行と７行が動的な制限情報に相当する。

【００６７】まず、シミュレーションが図７の第４行の制限を加えるシミュレーション時刻を設定する条件である時刻１０００に達し、この時の変数"system.filter2.mode"の値が8'hfaであるとすると、第２の論理回路削減手段５０５はインスタンス"system.filter2"に対応するモジュールの記述（図８）の変数"mode"を8'hfaに固定して第１の実施の形態と同様に論理を削減し、図４のように変換する。

【００６８】そして、シミュレーション手段５０６は、インスタンス"system.filter2"に対応するモデルとして図８の代わりに図４を用いてシミュレーションを続行する。

【００６９】次に、信号sys.modeの値が０から１に変化した場合、図７の第７行の制限に対し、制限を加えるシミュレーション時刻を設定する条件が成り立つため、制限情報に基づきシミュレーション対象（図６）からインスタンス"system.filter2"を取り除いたモデル（図９）を作成し、図８の代わりに図９を用いてシミュレーションを続行する。

【００７０】ただし、この時削除したインスタンスの出力値については、削除時の値を保持するように設定している。

【００７１】さらに、信号sys.modeの値が１から０に変化した場合、図７の第７行の制限に対し、制限を解除するシミュレーション時刻を設定する条件が成り立ち、実行中のシミュレーション対象を図９から図８に戻してシミュレーションを継続する。

【００７２】ただし、この時戻されるインスタンス"system.filter2"の内部状態は、削除時の値を用いる。

【００７３】なお、第２の実施の形態では制限を加えるシミュレーション時刻の設定を制限情報としてシミュレーション開始前に与えているが、論理回路のシミュレータにユーザインタフェースを付加して、シミュレーション途中に与えることも可能である。

【００７４】以上のように、本発明の第２の実施の形態によれば、請求項２、３に係る発明である論理回路のシミュレーション方法により、シミュレーションの対象となる論理回路から、目的とする検証に一定の期間不必要になる部分を、特定期間削除することにより、目的とする検証に必要なシミュレーションを高速に行なうことが可能となる。

【００７５】（第３の実施の形態）第３の実施の形態では、請求項４にかかる発明について図を用いて説明する。

【００７６】図１０は請求項４に係る論理シミュレーション手段の論理回路制限情報を抽出する手段の概略概念図である。

【００７７】図１０において１００１は解析対象のソフトウェア命令系列であるマシン語である。１０１１は論理回路に与えることができる制限情報である論理回路部可制限情報であり、１０１２は可制限情報を入力する可制限情報入力手段である。１００２は可制限情報入力手段１０１２からの情報より、論理回路の制限情報１００６を抽出する制限情報抽出手段である。

【００７８】図２３は第３の実施の形態に係るハードウェアの構成図の一例を示しており、図２３において、２３０１は処理されたあらゆる情報をみるためのディスプレイ装置、２３０２は設計者があらゆる情報や処理命令を入力するためのキーボード、２３０３はあらゆる処理を行う中央演算処理装置、２３０４は各情報を格納する記憶装置である。

【００７９】次に、例を用いて第３の実施の形態の論理シミュレーション手段を具体的に説明する。図１１はシミュレートされるプロセッサの命令セットの一例である。命令ＬＤはアドレスにより指定されるメモリ領域のデータからレジスタＡまたはレジスタＢにデータを転送するデータ転送命令である。命令ＳＴはレジスタＡまたはレジスタＢからアドレスにより指定されるメモリ領域にデータを転送するデータ転送命令である。命令ＭＯＶはソースで指定されるレジスタＡもしくはレジスタＢまたは定数値をデスティネーションで指定されるレジスタＡもしくはレジスタＢまたはレジスタＰＳＲ(プロセッサステータスレジスタ）に代入する算術演算命令である。レジスタＰＳＲの１ビット目は割込みに対するマスクビットになっており、このビットが１の場合は割込みを受け付けず、０の場合は割込みを受け付ける。スタート時はＰＳＲの１ビット目は１に設定されている。また、割込みを受け付けた時は自動的にＰＳＲの１ビット目に１が設定される。命令ＡＤＤはソースで指定されるレジスタＡもしくはレジスタＢまたは定数値とデスティネーションで指定されるレジスタＡまたはレジスタＢとを加算し、結果をデスティネーションで指定されているレジスタに代入する算術演算命令である。命令ＳＵＢはソースで指定されるレジスタＡもしくはレジスタＢまたは定数値をデスティネーションで指定されるレジスタＡ

11

またはレジスタBから減算し、結果をデスティネーションで指定されているレジスタに代入する算術演算命令である。命令MULはソースで指定されるレジスタAもしくはレジスタBまたは定数値とデスティネーションで指定されるレジスタAまたはレジスタBとを乗算し、結果をデスティネーションで指定されるレジスタに代入する算術演算命令である。命令CMPはソースで指定されるレジスタAもしくはレジスタBまたは定数値とデスティネーションで指定されるレジスタAまたはレジスタBとを比較し、等しければ比較結果フラグをセットし、等しくなければ比較結果フラグをリセットする比較命令である。比較結果フラグは次にCMP命令が実行されるまで保存される。命令BEQは、比較結果フラグがセットされていれば、ラベルで指定されている命令位置に処理をジャンプさせる条件分岐命令である。命令JMPはラベルで指定されている命令位置に処理をジャンプさせる無条件分岐命令である。命令RETIは割り込み信号が入力された後に処理される割り込み処理から、割り込み信号が入力される前に処理されていた命令位置に処理を戻す割り込み処理復帰命令である。命令NOPは何も処理を行わない。命令HALTは割り込みが入力されるまで何も行なわない。このプロセッサは1クロックに1命令実行することができる。

【0080】図12は、プロセッサ部と論理回路部とから成る被検証対象回路の例である。1201は図11の命令セットを持つプロセッサである。1202はプロセッサ1201に接続されているデータメモリである。1203は図2のハードウェア記述言語を用いて記述された論理回路である。1204、1205はセレクタであり、1206はインバータである。

【0081】図13は実行するプログラムの一例である。図13において一列が一命令を表しており、左からラベル、命令、デスティネーション、ソースの順番で記述されている。1301が通常実行されるプログラムであり、1302が割込み受付時に実行されるプログラムである。

【0082】図12より1201のプロセッサのAD[7]が1202のデータメモリと1203のフィルタへのアクセスを切り　えている。このため図12のプロセッサ1201からみたメモリマップと割込み情報より、図14の論理回路の可制限情報を作成することができる。1401がアクセスされるアドレスとアクセス条件、1402が1401へのアクセスが行われた時にアクティブになるモジュール、信号またはレジスタ変数を示している。1403は1401のアクセスが行われない場合、そのモジュール、信号またはレジスタ変数に対する処理を示している。1行目はアドレス0x80へのあらゆるアクセスが行われない場合、モジュールfilterが削除されることを示している。2行目は0x80のアドレスがフィルタの‘mode’レジスタ変数に

12

割り当てられていることを示している。3行目はアドレス0x80への読み出しアクセス時に‘filter.re’信号がアクティブになり、それ以外はこの信号が無視できることを、4行目はアドレス0x80への書き込みアクセス時に‘filter.we’信号がアクティブになり、それ以外はこの信号が無視できることを示している。5行目はPSRの1ビット目が0であれば、‘filter.sigovf’信号がアクティブになり、それ以外では無視できることを示している。

【0083】図26に1002の制限情報抽出手段の内部処理フローを示す。2601は割込みマスクの設定処理抽出手段であり、2602は通常プログラムの制限情報抽出手段であり、2603は割り込み処理プログラムの制限情報抽出手段である。

【0084】1302の通常プログラムが2601の割込みマスク設定処理抽出手段により、プロセッサの割込みはリセット状態ではマスクされているため、初期化時には割込みマスク状態であり、割込みマスクの解除としてラベルL8のプログラムアドレスが抽出される。次に2602の通常プログラムの制限情報抽出手段と図14の制限情報より、［0x80］への書き込みがラベルL1のプログラムアドレスが抽出される。さらに2603の割込み処理プログラムの制限情報抽出処理より、［0x10］からの読み出しとして、ラベルL11のプログラムアドレス、［0x80］からの読み出しとしてラベルL10のプログラムアドレス、［0x80］への書き込みとしてラベルL13のプログラムアドレスが抽出される。これらの結果、図15の制限情報が抽出される。

【0085】図15において、1500は制限を加える信号またはモジュール名である。1501は加える制限の種類であり、1502は信号を固定して用いる場合の信号値であり、1503は制限を加えるシミュレーション時刻を設定する条件であり、プロセッサのプログラムカウンタのプログラムアドレスが一致した場合で与えている。1504は制限を解除するシミュレーション時刻を設定する条件であり、プロセッサのプログラムカウンタのプログラムアドレスが一致した場合で与えている。

【0086】図15の制限情報の結果、図12の1203のフィルタはプロセッサ1201のプログラムカウンタがラベルL1のプログラムアドレスを実行するシミュレーション時刻から、プログラムカウンタがラベルL8のプログラムアドレスを実行するシミュレーション時刻までの間は図4に示す削減された論理回路となる。

【0087】以上のように、本発明の第3の実施の形態によれば、請求項4に係る発明である論理シミュレーション方法によって、プロセッサ部で実行されるソフトウェアに依存した論理削減により、ハードウェアとソフトウェアの相互作用を高精度に且つ高速にシミュレーションすることが可能となる。

【0088】（第4の実施の形態）第4の実施の形態で

13

は、請求項5に係る発明について図を用いて説明する。

【0089】図16は請求項5に係る論理シミュレーション装置の論理回路制限情報の生成および実行プログラム生成までの概略概念図である。

【0090】図16において、1601はコンパイラに入力される高級言語で記述されたソフトウェアコードであり、1611は1601のソフトウェアコードに対して検証対象を指定するソフトウェア指定情報であり、1621は論理回路に与えることができる制限情報である論理回路部可制限情報である。1612はソフトウェアの指定情報を入力する手段であり、1622は可制限情報を入力する可制限情報入力手段である。1602は1612のソフトウェア指定情報入力手段からと1622の論理回路の可制限情報より、プロセッサのソフトウェアプログラムをコンパイルするコンパイラである。1602はプロセッサが直接実行可能なマシン語1631と論理回路の制限情報1632を生成する。

【0091】図23は第4の実施の形態に係るハードウェアの構成図の一例を示しており、図23において、2301は処理されたあらゆる情報をみるためのディスプレイ装置、2302は設計者があらゆる情報や処理命令を入力するためのキーボード、2303はあらゆる処理を行う中央演算処理装置、2304は各情報を格納する記憶装置である。

【0092】次に、例を用いて第4の実施の形態の論理シミュレーション装置を具体的に説明する。

【0093】図17は、プロセッサ部と論理回路部とから成る被検証対象回路の例である。1701は図11の命令セットを持つプロセッサである。1702はプロセッサ1701に接続されているデータメモリである。1703、1704は図2のハードウェア記述言語を用いて記述された論理回路である。1705、1706はセレクタであり、1707はインバータ、1708は論理和である。

【0094】図18はコンパイル対象となる高級言語で記述されたプログラムの一例である。図18の例ではC言語で記述されている。

【0095】図17より1701のプロセッサのAD[7]が1702のデータメモリと1703と1704のフィルタF1、F2へのアクセスを切り替えている。AD[6]はフィルタF1とF2へのアクセスを切り替えている。このため図17のプロセッサからみたメモリマップと割込み情報より、図19の論理回路の可制限情報を作成することができる。1901がアクセスされるアドレスとアクセス条件、1902が1901へのアクセスが行われた時にアクティブになるモジュール、信号またはレジスタ変数を示している。1903は1901のアクセスが行われない場合、そのブロック、信号またはレジスタ変数に対する処理を示している。1行目は0xc0へのあらゆるアクセスが行われない場合、モジュー

14

ルF1が削除可能であることを、2行目は0x80へのあらゆるアクセスが行われない場合、モジュールF2が削除可能であることを示している。3行目は0xc0のアドレスがフィルタF1の‘mode’レジスタ変数に割り当てられていることを示している。4行目は0x80のアドレスがフィルタF2の‘mode’レジスタ変数に割り当てられていることを示している。5行目はアドレス0xc0への読み出しアクセス時に‘F1.re’信号がアクティブになり、それ以外はこの信号が無視できることを、6行目はアドレス0xc0への書き込みアクセス時に‘F1.we’信号がアクティブになり、それ以外はこの信号が無視できることを示している。7行目はアドレス0x80への読み出しアクセス時に‘F2.re’信号がアクティブになり、それ以外はこの信号が無視できることを、8行目はアドレス0x80への書き込みアクセス時に‘F2.we’信号がアクティブになり、それ以外はこの信号が無視できることを示している。9行目はPSRの1ビット目が0であれば、‘F1.sigovf’信号および‘F2.sigovf’がアクティブになり、それ以外では無視できることを示している。

【0096】図20はソフトウェアの検証範囲を指定するソフトウェア指定情報である。Fiset()関数を無視するという情報が記述されている。

【0097】図21は1602のコンパイラが図18、図19、図20の高級言語プログラム、論理回路の可制限情報、ソフトウェア指定情報より生成したマシン語であり、図22が制限情報である。図22において、2200は制限を加える信号またはモジュール名である。2201は加える制限の種類であり、2202は信号を固定して用いる場合の信号値であり、2203は制限を加えるシミュレーション時刻を設定する条件である。図20のソフトウェア指定情報に従って、1602のコンパイラは図18のC言語よりFiset()関数をコンパイル対象から削除する。その結果、アドレス0xc0へのアクセスが全く行われなくなり、図19の1行目の可制限情報より、図22の1行目の制限情報が生成される。

【0098】図22の制限情報の結果、モジュールF1は削除され、図12の1203のフィルタはプログラムカウンタがラベルL1のプログラムアドレスになったシミュレーション時刻以降、図9に示す削減された論理回路となる。

【0099】以上のように、本発明の第4の実施の形態によれば、請求項5に係る発明である論理シミュレーション装置によって、検証目的に応じて、プロセッサ部を制御するソフトウェアを指定し、論理回路部に与えることができる制限情報から、プロセッサ部を制御するプログラム命令系列を生成し、且つ論理回路部の制限情報を生成するコンパイラを備え、生成された制限情報に基づ

き論理を削減することにより、シミュレーションを高速化することが可能となる。

【0100】以上説明した様に、請求項1から請求項5に対応する第1から第4の実施の形態の発明によれば、論理回路に検証目的に応じた制限情報を与えることにより、論理回路を削減することが可能である。また、削減された論理回路に対してシミュレーションを行なうことにより、高速に論理検証を行なうことが可能である。また、プロセッサ部とプロセッサにより制御される論理回路部を持ち、ソフトウェアプログラムコードと論理回路部が相互に作用するシステム全体のシミュレーションにおいて、論理回路部の情報と、ソフトウェアの解析により、検証を行なうソフトウェアから論理回路部の制限情報を作成、論理を削減することにより高速なシミュレーションが可能である。

【0101】

【発明の効果】請求項1記載の論理削減装置によれば、検証目的に応じた制限情報を与え、その制限情報に基づき論理回路から構成される被検証対象回路を削減する方法により、被検証対象回路を入力し、検証目的に応じた制限情報を入力し、入力した制限情報に基づき、入力された被検証対象回路を削減することができる。このため、高速に論理シミュレーションを行なうことができる。

【0102】請求項2記載の論理シミュレーション方法によれば、削減された論理回路の動作をシミュレートする方法により、論理シミュレーションを高速化することができる。

【0103】請求項3記載の論理シミュレーション装置によれば、検証目的に応じた制限情報の各々にシミュレーション時刻に依存した制限情報を与え、シミュレーション時刻に依存した制限情報に基づき、シミュレーション時刻を分割し、分割されたシミュレーション時刻毎に被検証対象回路の論理を削減し、論理シミュレートする方法により、全体の論理シミュレーションを高速化することができる。

【0104】請求項4記載の論理シミュレーション装置によれば、プロセッサ部を制御するソフトウェアの命令動作を解析し、論理回路部に与えることができる制限情報を入力し、この制限情報とソフトウェアの命令動作の解析結果から、論理を削減する制限情報を抽出し、抽出された制限情報に基づき論理を削減する方法により、論理シミュレーションを高速化することができる。

【0105】請求項5記載の論理シミュレーション装置によれば、検証目的に応じて、プロセッサ部を制御するソフトウェアを指定し、論理回路部に与えることができる制限情報から、プロセッサ部を制御するプログラム命令系列を生成するとともに論理回路部の制限情報を生成し、生成された制限情報に基づき論理を削減する方法により、シミュレーションを高速化することができる。

【図面の簡単な説明】

【図1】本発明の第1の実施の形態における構成を示すブロック図である。

【図2】本発明の第1、第2、第3および第4の実施の形態において例として用いる論理回路をハードウェア記述言語を用いて記述した図である。

【図3】本発明の第1の実施の形態において例として用いる制限情報を示す図である。

【図4】本発明の第1および第3の実施の形態において論理の削減結果として得られる論理回路を表す図である。

【図5】本発明の第2の実施の形態における構成を示すブロック図である。

【図6】本発明の第2の実施の形態において例として用いる論理回路の一部をハードウェア記述言語を用いて記述した図である。

【図7】本発明の第2の実施の形態において例として用いる制限情報を示す図である。

【図8】本発明の第2、第4の実施の形態において静的な論理の削減結果として得られる論理回路を表す図である。

【図9】本発明の第2の実施の形態において動的な論理の削減結果として得られるシミュレーション対象を表す図である。

【図10】本発明の第3の実施の形態における制限情報抽出の概略概念のブロック図である。

【図11】本発明の第3の実施の形態におけるプロセッサの命令セットの例を示す図である。

【図12】本発明の第3の実施の形態におけるプロセッサ部と論理回路部とから成る被検証対象回路の例を示すブロック図である。

【図13】本発明の第3の実施の形態において実行するプログラムを示す図である。

【図14】本発明の第3の実施の形態における論理回路の可制限情報を示す図である。

【図15】本発明の第3の実施の形態において抽出された論理回路の制限情報を示す図である。

【図16】本発明の第4の実施の形態における論理回路制限情報の生成および実行プログラム生成の概略概念のブロック図である。

【図17】本発明の第4の実施の形態におけるプロセッサ部と論理回路部とから成る被検証対象回路の例を示すブロック図である。

【図18】本発明の第4の実施の形態におけるコンパイル対象となる高級言語で記述されたプログラムの一例を示す図である。

【図19】本発明の第4の実施の形態における論理回路の可制限情報を示す図である。

【図20】本発明の第4の実施の形態におけるソフトウェアの検証範囲を指定するソフトウェア指定情報を示す

図である。

【図21】本発明の第4の実施の形態においてコンパイラが生成したマシン語を示す図である。

【図22】本発明の第4の実施の形態においてコンパイラが生成した論理回路の制限情報を示す図である。

【図23】本発明の第1、第2、第3および第4の実施の形態におけるハードウェアの構成図の一例を示したブロック図である。

【図24】本発明の第1の実施の形態において論理回路の削減方法を示すフロー図である。

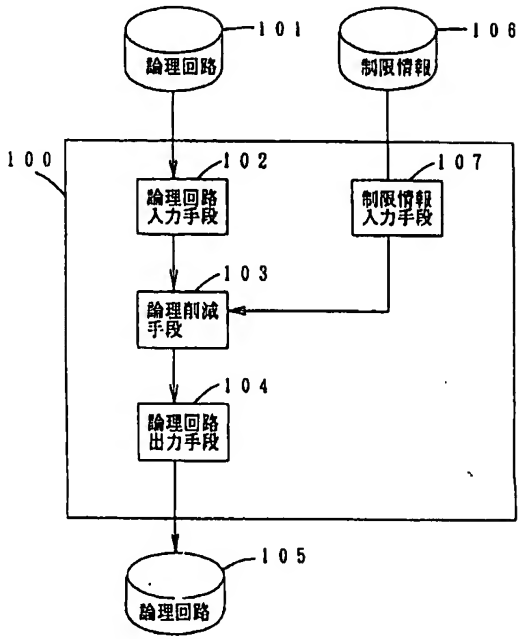
【図25】本発明の第2の実施の形態においてシミュレーション対象を変換する方法を示すフロー図である。

【図26】本発明の第3の実施の形態において制限情報抽出手段1002の概念フロー図である。

【符号の説明】

100 論理回路の変換装置
102 論理回路入力手段
103 論理削減手段
*107 制限情報入力手段
500 論理回路のシミュレータ
502 論理回路入力手段
503 静的情報を用いる論理削減手段
505 動的情報を用いる論理削減手段
506 シミュレーション手段
508 制限情報入力手段
509 シミュレーション時刻分割手段
1002 制限情報抽出手段
1003 論理回路の制限情報を抽出する制限情報抽出手段
1012 可制限情報を入力する可制限情報入力手段
1602 論理回路制御情報も生成するコンパイラ
1612 ソフトウェアの指定情報を入力するソフトウェア指定情報入力手段
1622 可制限情報を入力する可制限情報入力手段

【図1】



【図2】

```
 1 module filter(clk,reset,re,we,rdata,wdata,sigin,sigout,sigovf);
 2         input   clk;
 3         input   reset;
 4         input   re;
 5         input   we;
 6         output  [7:0] rdata;
 7         input   [7:0] wdata;
 8         input   [7:0] sigin;
 9         output  [7:0] sigout;
10         output  sigovf;
11
12         reg     [7:0] mode;
13         reg     [7:0] sighld1;
14         reg     [8:0] sighld2;
15         reg     [8:0] sighld3;
16         reg     [8:0] sighld4;
17
18         always @(posedge clk) begin
19                 if ( reset ) begin
20                         mode = 7'h0;
21                 end
22                 else begin
23                         mode <= we ? wdata : mode;
24                 end
25         end
26
27         assign rdata = re ? mode : 8'hzz;
28
29         always @(posedge clk) begin
30                 sighld1 <= mode[0] ? 8'h0 : mode[4] ? sigin[7:0]
31                                 : { 1'b0,sigin[7:1]};
32                 sighld2 <= sighld1 + mode[1] ? 8'h0 : mode[5] ?
33                                 sigin[7:0] : { 1'b0,sigin[7:1]};
34                 sighld3 <= sighld2[7:0] + mode[2] ? 8'h0 : mode[6] ?
35                                 sigin[7:0] : { 1'b0,sigin[7:1]};
36                 sighld4 <= sighld3[7:0] + mode[3] ? 8'h0 : mode[7] ?
37                                 sigin[7:0] : { 1'b0,sigin[7:1]};
38         end
39
40         assign sigout = sighld4[7:0];
41         assign sigovf = sighld2[8] | sighld3[8] | sighld4[8];
42 endmodule
```

【図3】

```
300          301     302          Fiset( )      ignore

filter.mode  fix     8'hfa -- 303
filter.re    fix     1'b0  -- 304
filter.sigovf ignore         305
```

【図20】

【図4】

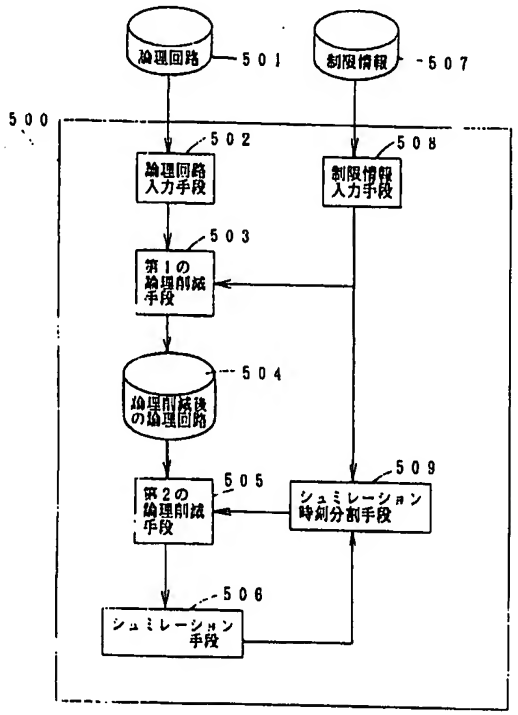
```
 1  module filter(clk,reset,re,we,rdata,wdata,sigin,sigout,sigovf);
 2          input   clk;
 3          input   reset;
 4          input   re;
 5          input   we;
 6          output  [7:0] rdata;
 7          input   [7:0] wdata;
 8          input   [7:0] sigin;
 9          output  [7:0] sigout;
10          output  sigovf;
11
12          reg     [7:0] mode;
13          reg     [7:0] sighld1;
14          reg     [8:0] sighld2;
15          reg     [8:0] sighld3;
16          reg     [8:0] sighld4;
17
18          assign rdata = 8'hzz;
19
20          always @(posedge clk) begin
21                  sighld1 <= 8'h0;
22                  sighld2 <= sighld1 + { 1'b0,sigin[7:1]};
23                  sighld3 <= sighld2;
24                  sighld4 <= sighld3 + { 1'b0,sigin[7:1]};
25          end
26
27          assign sigout = sighld4;
28  endmodule
```

【図7】

```
700                 701     702      703          704

system.filter1.mode     fix     8'hfa   initial
system.filter1.re       fix     1'b0    initial
system.filter1.sigovf   ignore          initial
system.filter2.mode     fix     hold    #1000
system.filter2.re       fix     1'b0    initial
system.filter1.sigovf   ignore          initial
system.filter2          remove          posedge system.mode negedge system.mode
```

【図5】



【図14】

1. 0×80	filter	remove
2. (0×80)	filter.mode	
3. (0×80).read	filter.re	ignore
4. (0×80).write	filter.we	ignore
5. PSR & 0×00	filter.sigovf	ignore
≀	≀	≀
1401	1402	1403

【図6】

```
module system;
        .
        .
        .


filter  filter1( .clk(clk), .reset(reset), .re(re1),
                 .we(we1), .rdata(rdata), .wdata(wdata),
                 .sigin(sigin1), .sigout(sigout1), .sigovf());

filter  filter2( .clk(clk), .reset(reset), .re(re2),
                 .we(we2), .rdata(rdata), .wdata(wdata),
                 .sigin(sigin2), .sigout(sigout2), .sigovf());

assign  sigout = mode ? sigout1 : sigout2;

        .
        .
        .
endmodule;
```

【図9】

```
module system;
        .
        .
        .


filter  filter1( .clk(clk), .reset(reset), .re(re1),
                 .we(we1), .rdata(rdata), .wdata(wdata),
                 .sigin(sigin1), .sigout(sigout1), .sigovf());

assign rdata = 8'hzz;
assign sigout2 = 8'h28;

assign  sigout = mode ? sigout1 : sigout2;

        .
        .
        .
endmodule;
```

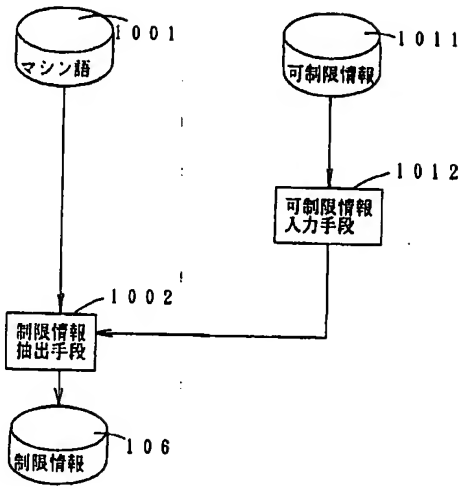
【図8】

```
 1 module filter(clk,reset,re,we,rdata,wdata,sigin,sigout,sigovf);
 2         input   clk;
 3         input   reset;
 4         input   re;
 5         input   we;
 6         output  [7:0] rdata;
 7         input   [7:0] wdata;
 8         input   [7:0] sigin;
 9         output  [7:0] sigout;
10         output  sigovf;
11
12         reg     [7:0] mode;
13         reg     [7:0] sighld1;
14         reg     [8:0] sighld2;
15         reg     [8:0] sighld3;
16         reg     [8:0] sighld4;
17
18         always @(posedge clk) begin
19                 if ( reset ) begin
20                         mode = 7'h0;
21                 end
22                 else begin
23                         mode <= we ? wdata : mode;
24                 end
25         end
26
27         always @(posedge clk) begin
28                 sighld1 <= mode[0] ? 8'h0 : mode[4] ? sigin[7:0]
29                         : { 1'b0,sigin[7:1]};
30                 sighld2 <= sighld1 + mode[1] ? 8'h0 : mode[5] ?
31                         sigin[7:0] : { 1'b0,sigin[7:1]};
32                 sighld3 <= sighld2[7:0] + mode[2] ? 8'h0 : mode[6] ?
33                         sigin[7:0] : { 1'b0,sigin[7:1]};
34                 sighld4 <= sighld3[7:0] + mode[3] ? 8'h0 : mode[7] ?
35                         sigin[7:0] : { 1'b0,sigin[7:1]};
36         end
37
38         assign sigout = sighld4[7:0];
39 endmodule
```

【図15】

1. filter.mode	fix	hold	PC=L1	PC=L8
2. filter.sigovf	ignore		initial	PC=L8
3. filter.re	fix	hold	initial	PC=L8
⟨	⟨	⟨	⟨	⟨
1500	1501	1502	1503	1504

【図10】

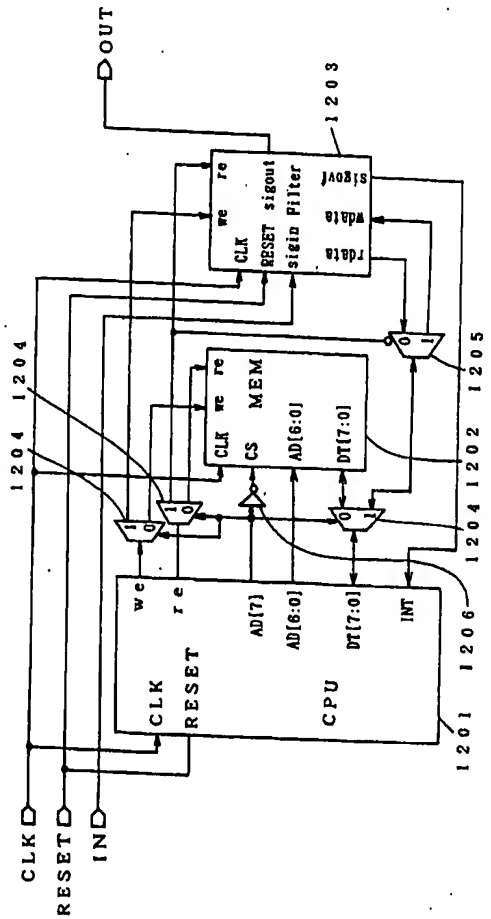


【図11】

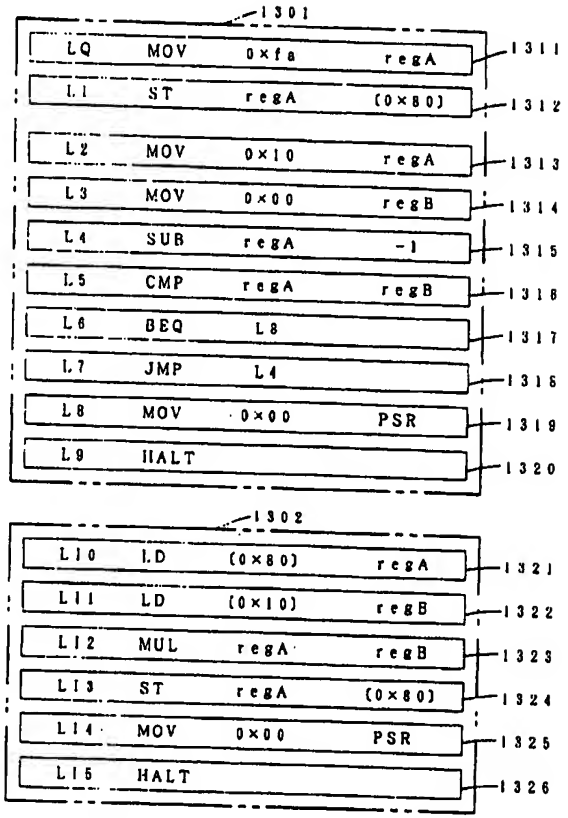
1101

命 令	デスティネーション	ソース
LD	regA or regB	アドレス
ST	regA or regB	アドレス
MOV	regA or regB or PSR	regA or regB or 定数値
ADD	regA or regB	regA or regB or 定数値
SUB	regA or regB	regA or regB or 定数値
MUL	regA or regB	regA or regB or 定数値
CMP	regA or regB	regA or regB or 定数値
BEQ	ラベル	
JMP	ラベル	
RETI		
NOP		
HALT		

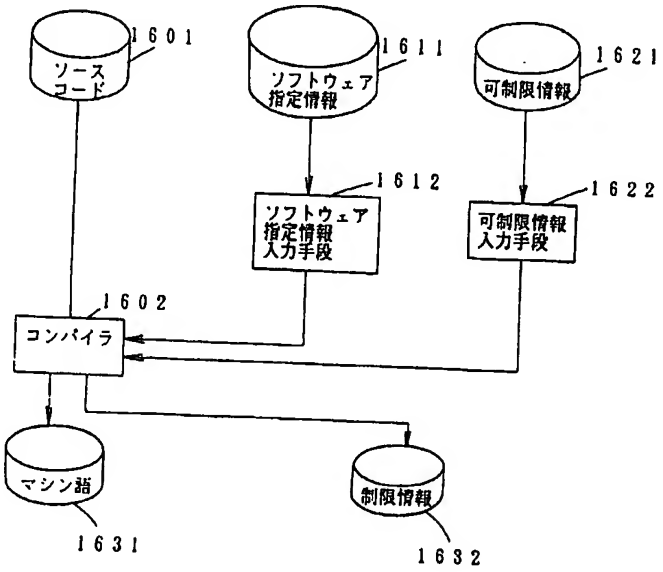
【図12】


OUT
1203
re
we
CLK
RESET
sigout
sigin
Filter
sigovf
wdata
rdata
MEM
CS
AD[6:0]
DT[7:0]
1204
1202
1205
1206
1201
CPU
INT
CLK
RESET
IN

【図13】


1301
L0 MOV 0×fa regA 1311
L1 ST regA (0×80) 1312
L2 MOV 0×10 regA 1313
L3 MOV 0×00 regB 1314
L4 SUB regA -1 1315
L5 CMP regA regB 1316
L6 BEQ L8 1317
L7 JMP L4 1318
L8 MOV 0×00 PSR 1319
L9 HALT 1320
1302
L10 LD (0×80) regA 1321
L11 LD (0×10) regB 1322
L12 MUL regA regB 1323
L13 ST regA (0×80) 1324
L14 MOV 0×00 PSR 1325
L15 HALT 1326

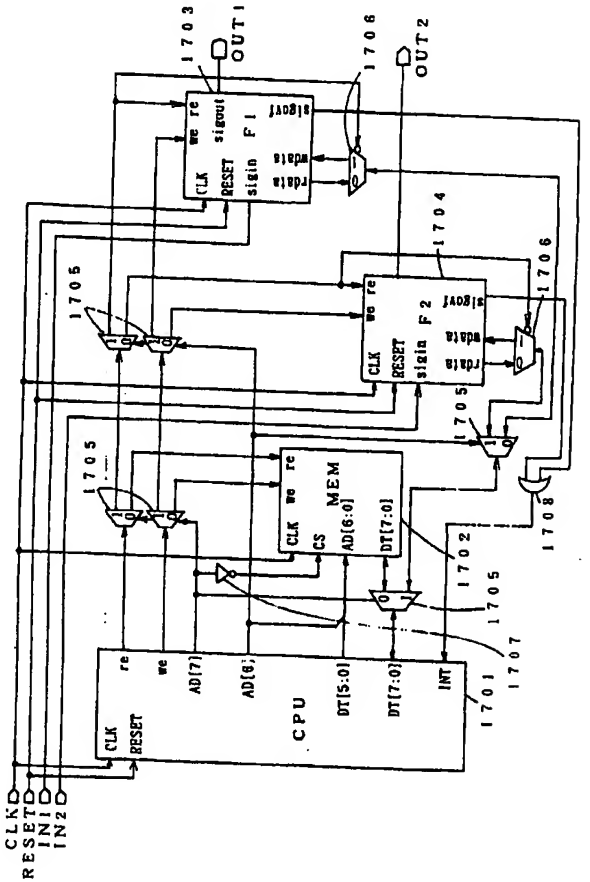
【図16】



【図19】

1. 0×C0	F1	remove
2. 0×80	F2	remove
3. (0×C0)	F1.mode	
4. (0×80)	F2.mode	
5. (0×C0).read	F1.re	ignore
6. (0×C0).write	F1.we	ignore
7. (0×80).read	F2.re	ignore
8. (0×80).write	F2.we	ignore
9. PSR & 0×00	F1.sigovf, F2.sigovf.	ignore
1901	1902	1903

【図17】



【図18】

```
F1set(void)
{
        char a=0×fa
        memset(0×c0, a, 1);
}
F2set(void)
{
        char a=0×bc
        memset(0×80, a, 1);
}

main( )
{
        F1set( );
        F2set( );
        for(a=0×100; a!=0 ;a--
}
```

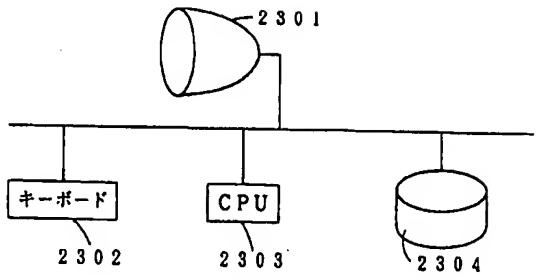
【図21】

L0	MOV	0×bc	regA
L1	ST	regA	(0×80)
L2	MOV	0×100	regA
L3	MOV	0×00	regB
L4	SUB	regA	-1
L5	CMP	regA	regB
L6	BEQ	L8	
L7	JMP	L4	
L8	HALT		

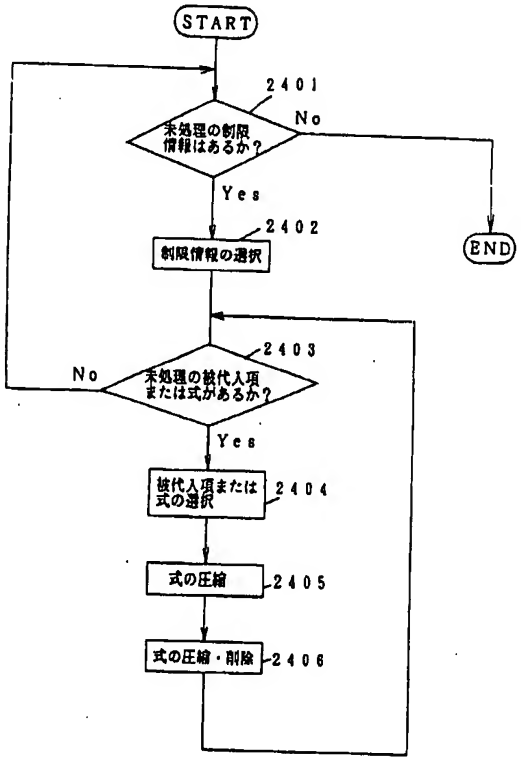
【図22】

F1	remove		
F2.mode	fix	hold	PC=L1
F2.sigovf	ignore		initial
F2.re	fix	hold	initial
2200	2201	2202	2203

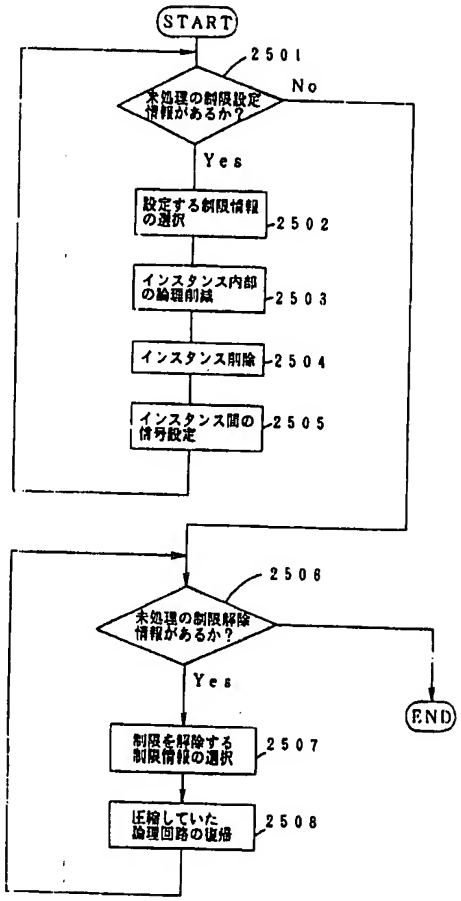
【図23】



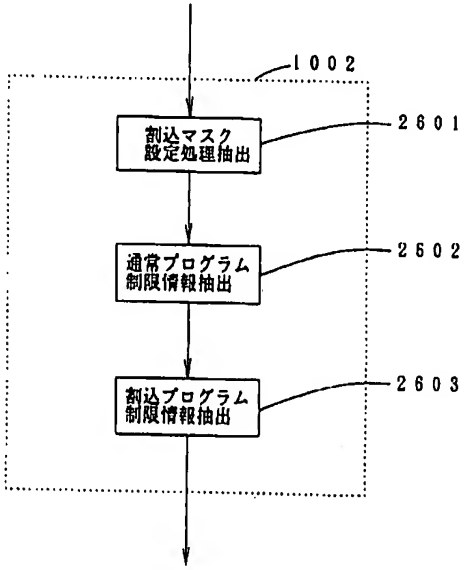
【図24】


START
2401
未処理の制限情報はあるか？
No
Yes
END
2402
制限情報の選択
2403
No
未処理の被代入項または式があるか？
Yes
被代入項または式の選択
2404
式の圧縮
2405
式の圧縮・削除
2406

【図25】


START
2501
未処理の制限設定情報があるか？
No
Yes
設定する制限情報の選択
2502
インスタンス内部の論理削減
2503
インスタンス削除
2504
インスタンス間の信号設定
2505
2506
未処理の制限解除情報があるか？
Yes
END
制限を解除する制限情報の選択
2507
圧縮していた論理回路の復活
2508

【図26】


1002
割込マスク
設定処理抽出
2601
通常プログラム
制限情報抽出
2602
割込プログラム
制限情報抽出
2603